An Architecture and Key Management Approach for Maintaining Privacy in Location Based Group Services

Y. Sun[1], P. Liu[1], P. Kermani[2], T. F. La Porta[1]
1- Networking and Security Research Center, Penn State University
yasun@cse.psu.edu, pliu@ist.psu.edu, tlp@cse.psu.edu
2- IBM T.J. Watson
parviz@us.ibm.coml

Abstract

Location based services are becoming increasingly important to the success and attractiveness of next generation wireless systems. Service providers will use location information to introduce new services and greatly enhance many existing services. Maintaining location privacy is an important requirement that must be met for these services to be widely deployed. It is a challenge to maintain location privacy while still providing the flexible access to location information required to enable a rich set of location based services. In this paper we define a high-level architecture for providing LBS and classify services according to several basic criteria. To support these services we propose a hierarchical key distribution method. Four methods are proposed to deliver hierarchical location information while maintaining privacy. We evaluate the efficiency of the system in terms of message delivery and key management overhead.

1. Introduction

One of the most desirable classes of service expected to be offered in future wireless systems are so called Location-based services (LBS). Service providers envision providing many new services based on user location as well as augmenting many existing services with location information. LBS are unique to mobile environments, and are especially powerful in a wireless setting.

One concern with LBS is the protection of user privacy with respect to their location. Consider the case of an account manager traveling to the location of a potential customer. While in a new town the account manager can use LBS to find local restaurants, gas stations, pharmacies, etc. However, the account manager would not want competitors to access their location.

Great effort has been given to protecting user identity and location with basic cellular communication services. This protection must not be compromised with the addition of LBS without permission of the user.

A natural tension arises when attempting to protect user privacy while building a system that allows for flexible use of location information. Flexibility is essential so that new services may be introduced and modified quickly. However, providing flexible access to location information will weaken user privacy if care is not taken.

We propose a solution based on the following philosophy: access to location information is controlled by the user. The user defines a group of other entities that are allowed to access its location information. This group may include other end users and servers in the network that use the location information to provide intelligent services. To increase flexibility, our system supports the hierarchical coding of location information so that different group members may have access to only the granularity of location information required to deliver the appropriate service. For example, location information may be coded as (country, state, city, address, coordinates). This provides the attractive characteristic that application servers may use location information as a building block for their own services; i.e., not every application must implement a method of determining user location.

The solution is based on providing keys to the group members that allow them to decrypt location information for their use. A main challenge is to efficiently distribute these keys in the face of changing group membership while supporting hierarchical location information.

In this paper we define an architecture for providing LBS and classify services according to several basic criteria. We propose four methods for delivering hierarchical location information while maintaining privacy. To support these services we propose a hierarchical key distribution method. We evaluate the efficiency of the system in terms of message delivery and key management overhead.

The remainder of the paper is organized as follows. In Section 2 we briefly discuss related work. In Section 3 we present our service model and an overview of our system. In Section 4 we present the details of hierarchical coding and location information delivery. In Section 5 we present the key distribution mechanism. In Section 6 we evaluate the efficiency of the system. In Section 7 we discuss trade-offs of the design, and we conclude in Section 8.

2. Related work

We briefly introduce the related work in two areas: Location based services (LBS) and privacy and group key management.

2.1. Location based services and privacy

Location-based services and applications have been emerging with the rapid development of indoor and outdoor positioning technologies such as GPS, cell-phones and RFID. As specified in 3G standards, LBS systems include user equipment, a positioning sub-system, location server, requester and client [1]. There are also a set of IETF standards and drafts to support LBS [2].

Parlay/OSA provides a standardized, extendable and scalable interface that enables rapid creation of telecommunication services which can be used to build integrated location-based services [3].

There are concerns on how to protect location privacy while still providing the benefits of LBS. Previous work can be classified into two different approaches: solutions based on privacy policies [2, 4, 5], and those based on a location anonymizer [6-10]. The latter provides fuzzy location to make the identification of a device un-linkable to its location. The privacy protection solution depends on the mutual trust between the user and the service provider.

2.2. Group key management schemes

One important aspect of group communication security is access control, which can be achieved by encrypting the communication content using a secret key known to all group members. Many group key management protocols have been proposed to support these solutions. Solutions may be classified based on the existence of a Key Distribution Center, who is responsible for the key generation, and whether key management and distribution are centralized or distributed [11].

In a centralized system, there is only one entity controlling the group; protocols for these systems are often based on logical key trees, e.g. Logical Key Hierarchy (LKH) [12,13], One-way Function Tree (OFT)[14], or Efficient Large-Group Key (ELK)[15]. The best solutions appear to be those using a hierarchical tree of key-encrypting keys. They achieve good overall results without compromising any aspects of security. We extend LKH to support multiple level location information encryption. Our extensions can be modified to adapt to other logical key trees based schemes.

3. Solution overview

In this section we describe the service architecture, give an example of location information dissemination, briefly discuss hierarchical coding of location information, and illustrate two simple services using this architecture.

3.1. System architecture

The architecture we propose is shown in Figure 1. This architecture includes the basic functions required to provide a LBS and does not imply a physical implementation or deployment.

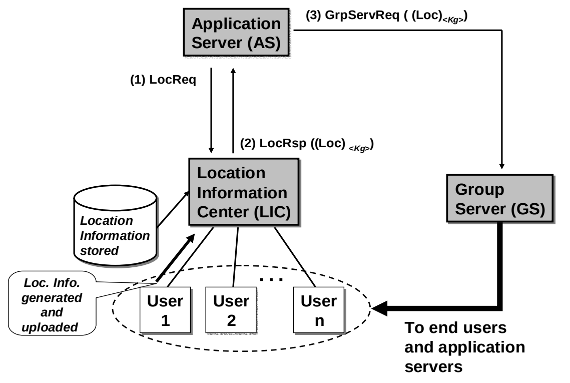
Application Server (AS)
(3) GrpServReq ((Loc)<Kg>)
(1) LocReq
(2) LocRsp ((Loc) <Kg>)
Location Information Center (LIC)
Group Server (GS)
Location Information stored
Loc. Info. generated and uploaded
User 1
User 2
. . .
User n
To end users and application servers

Figure 1. Location based services architecture

The user device may generate or assist in generating its own location information, and may receive the location of other end users as part of a service. The Location Information Center (LIC) is a network server that stores and potentially processes location information. The application servers (AS) provide the end service which uses the location information. The group server (GS) manages the groups that are to receive the location information including both end users and network servers. These functions may be combined in various ways in a network deployment. The exact responsibilities of each element are dependent on the following system characteristics:

- **WHERE location information is generated -** If the information is generated by the end device, network servers may be precluded from accessing it; if it is generated by the network, a LIC must be present.

- **WHERE location information is stored -** The information may be stored in the end device, network, or not at all. If the information is generated by the network it is most natural to store it in the LIC.
- **HOW information is accessed -** Information may be pushed at various intervals or pulled on-demand. In the former case, it is most efficient to disseminate the information from the network.
- **with WHOM the location information is shared** - Depending on the service, information may be shared with other end users, network servers, or both. The *location information group* is comprised of the end users and network servers that have access to the location information.

It is important to note that when information is stored in the LIC, it is not necessarily accessible by the LIC. For example, location information may be generated at the end device, perhaps using GPS, encrypted, and then uploaded to the LIC. In this case, this LIC will store the information and help disseminate it, thus relieving the end device of this burden, but will not be able to add value to any service. Alternatively, the LIC may be a member of the location information group, and be able to decrypt the information and process it, thus adding value to the services. In the case, the LIC must have a trusted relationship with the end user.

One important aspect of the system is the *hierarchical coding of location information*. We present the details of the delivery of hierarchical location information in Section 4; here we provide a brief overview to illustrate its use in the system. When location information is generated, for example using GPS or TDOA techniques, it will typically be in a *(x,y)* coordinate format. This information may then be processed to produce a set of information of differing granularity, for example {country, state, city, address, *(x,y)*}. In general, location information can be processed into *c* classes. The information may then be given out according to its class. This will increase flexibility because an end user may approve sharing of only the granularity of location information that they desire, and application servers will only have access to the information required to provide their service.

3.2. Location information dissemination

Figure 1 shows a representative example of location information dissemination. In this example, location information is generated by the end device and uploaded into the LIC (message 1). The application server requests the location information from the LIC on-demand (message 2), and sends it to the group server from which it is disseminated to the end users (message 3). This way, the application server is only concerned with triggers to disseminate the information, and the group server is responsible for managing group membership. In an actual deployment, these functions may be combined for the most efficient operation.

Consider the case in which only the end users are members of the location information group, i.e., the LIC and application servers cannot decrypt the location information. In this case, the LIC simply acts as network storage for the information.

If the LIC is a member of the location information group, it may decrypt the information from the end device, and process it to produce the hierarchy of location information. As discussed in Section 4, there are several ways in which this information may then be disseminated.

Finally, consider now that certain application servers are members of the location information group. These application servers may add their own processing to further increase the intelligence of the services they provide. By supporting hierarchical coding of the location information, they may only be allowed to access the level of information they required for their service.

The flow in Figure 1 is easily modified to support scenarios in which no LIC is present, or in which the location information is pushed to the application servers at a specified frequency or in response to a specified event.

3.3. Service examples

In this section we describe a location-based instant messaging service and the impact of the different deployment options discussed in Sections 3.1-3.2 on the operation of the service. The service works as follows. In addition to traditional instant messaging (IM) alerts when a member of a buddy list becomes on-line, and alert member is sent to subscribers when a buddy is within a certain proximity. The service is comprised of an IM system which requests the location information of its subscribers from the LIC.

We first consider the case in which only the end users and LIC are members of the location information group, i.e., the IM is not a member of the location information group. In this case the IM server plays the role of the application server in Figure 1 and requests location information from the LIC periodically. The LIC provides the encrypted information to the IM server, which sends it to the group server to be disseminated to all members of the buddy list. The end devices of these subscribers decrypt the information, compare it to their own location, and generate an alert locally if a buddy is within proximity of the device.

This implementation of the service has two interesting characteristics. First, the end user has complete control over which buddies know its location information; it may decide to limit the knowledge of its location to a subset (or none) of its buddies. Second, the value added service of proximity detection is implemented primarily in the end device; the IM server is simply facilitating the delivery of the information.

Now consider the case in which the IM server is a member of the location information group. In this case, when the IM receives the location information from the LIC, it will decrypt the information, compare it to the location of the other members of the buddy list, and execute directed alerts to only those members that are within proximity of each other. This implementation has three important characteristics. First, the end user must trust the IM server to only disclose its location information to approved buddies. Second, the delivery of alerts is more efficient than in the end user-based system discussed above. And third, the IM server is able to provide increased value to the service without placing a burden on the end device.

4. Delivery of hierarchical location information

In this section we discuss four methods of delivering hierarchical location information. Each method has trade-offs in terms of message delivery efficiency and group management complexity. We discuss the key distribution protocol in detail in Section 5, and evaluate the performance of these methods in Section 6.

4.1. Coding by information class

In this method, each class of location information, *c*, is encrypted with its own key, K_c, and location information groups are assigned based on a single class of information. With this system, a user requiring all *c* levels of location information will join all *c* groups and receive each piece of information independently of the others.

This method provides flexibility in that users will only receive the location information they require. However, it requires the management of *c* groups, and if users require more than one class of information, they must join more than one group. Each user will receive *c* multicast messages; one for each group to which is belongs.

4.2. Coding by group

In this method, location information is encrypted according to location information groups. Each location information group has access to a fixed set of classes of location information and has its own group key, K_{gi}. Formally, each group, *i*, receives location information in the following format:

Group 1 message format: $(\mathrm{Loc}_1)_{<K_{g1}>}$;
Group 2 message format: $(\mathrm{Loc}_1,\mathrm{Loc}_2)_{<K_{g2}>}$;
Group *i* message format: $(\mathrm{Loc}_1,\mathrm{Loc}_2,\ldots,\mathrm{Loc}_i)_{<K_{gi}>}$;
Group *c* message format: $(\mathrm{Loc}_1,\mathrm{Loc}_2,\ldots,\mathrm{Loc}_c)_{<K_{gc}>}$;

In this method each user must only join a single group and will only receive a single message that contains all location information at and below the level of their permission. The number of groups that must be managed is *c*.

4.3. Nested coding

In this method location information is encrypted using a nested hierarchy. End users belong to a location information group according to the maximum level of information they require. Each group has a group key $K_{gi.}$ Formally, the information is disseminated with a message format $\{\mathrm{Loc1},\{\mathrm{Loc2},\ldots,\{\mathrm{Loc}i,\ldots\{\mathrm{Loc}c\}_{<K_{gc}>}\}\ldots_{<K_{gi}>}\}\ldots_{<K_{g2}>}\}_{<K_{g1}>}\}$.

With this method only a single group is managed for message delivery. Each user receives the same information regardless of their class. The distinction between classes is based solely on the possession of different keys.

4.4. Flat coding

This scheme assumes a flat relationship among sub-groups. Each member of a location information group receives all of the information at or below the maximum level it requires. The message format for location information delivery is $\{\{\mathrm{Loc1}\}_{<K_{g1}>},\{\mathrm{Loc2}\}_{<K_{g2}>},\ldots,\{\mathrm{Loc}i\}_{<K_{gi}>},\ldots,\{\mathrm{Loc}c\}_{<K_{gc}>}\}$.

Using this scheme, each user must join only a single group. All users receive the same messages regardless of their class. As with nested encoding, groups are differentiated by the keys they maintain.

5. Group key management

As is evident, a main element of this system is the ability to distribute keys to the appropriate group members. The key distribution protocol must accommodate the hierarchical coding methods discussed in Section 4. The efficiency of the keying protocol will vary depending on group size and group dynamics. It is natural to apply group key management protocols used for multicast communication or collaborative systems to this problem.

Within the IETF there are several standards or draft standards that address key management for groups. These include the Group Domain of Interpretation (GDOI -

RFC3547) [16], GSAKMP [17], and MIKEY (RFC3830) [18]. Table 1 compares these three protocols for a small set of important characteristics.

Table 1. Keying Characteristics

Function	GDOI	GSAKMP	MIKEY
Registration	Y	Y	Y
Re-keying	Y	Y	N
Round-trip time	3.4-5.0	1.5-2.5	0.5-1.0
Hierarchy	N	Y	N

We select MIKEY as the basis of our protocol for two reasons. First, it is a lightweight protocol that can establish group keys with a low round trip delay. Second, MIKEY has been adopted by the 3GPP MBMS group for key management for 3G multimedia services [19, 20].

There are two limitations of MIKEY. First, it does not support a hierarchy of group key servers which may be important to scale the solution. Second, it does not support re-keying which is essential because group members may leave or join, and thus new keys must be established. We address these limitations in the following subsections after providing a brief overview of MIKEY below.

5.1. Overview of MIKEY

MIKEY uses a three level key management structure to distribute group keys to the clients. The User Key (MUK), is a point-to-point key between the multicast server and each client. This key is used by the key server to authenticate each client. The Service Key (MSK) is shared between the multicast server and all group members. The Traffic Key (MTK) is also shared between the key server and all group members. The MUK is used to protect the distribution of the MSK and the MSK is used to protect the distribution of MTK. The MTK is used to protect the data exchanged between members of the group.

The MSK is delivered to the group members after they are authenticated. Delivery of this key is point-to-point through push, pull or push solicited pull. The MTK traffic key delivery can utilize a multicast mechanism.

Because MIKEY does not support re-keying, each time a group member joins or leaves, a new MSK must be established with all group members, and a new MTK must be multicast to the entire group. Therefore, the overhead for re-keying is *O(N)*, and is not scaleable to large groups.

5.2. MIKEY and LKH

We apply LKH to MIKEY to improve scalability with re-keying. The basic LKH protocol uses a balanced tree to represent a logical key tree.

Figure 2 shows an example of a balanced key tree. In this Figure, the root key, K_g, is the MTK and is shared by all users. The children of the root, and the subsequent children down each branch of the tree, may be used to facilitate re-keying.

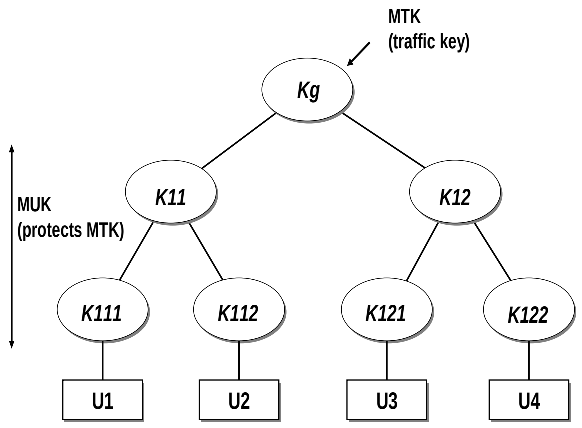

Figure 2. LKH hierarchy

Each user owns the keys on the path from the leaf node to the root of the tree. When a user joins or leaves, all the keys on the path have to be changed to maintain backward and forward security. For a join or leave event the communication overhead is *O(log N).*

For example, consider the case of user 2 (U2) leaving the group. To re-key U3 and U4, the new group key, K_g*, is multicast down this branch and encrypted with K_{12}. Before U1 can be re-keyed, K_{11} must be changed. The new K_{11}, $K_{11}*$, is distributed to U1 encrypted with K_{111}, a key it shares only with the key server. Once U1 has $K_{11}*$, it is sent K_g* directly by the key server.

Now consider the case of U2 re-joining the group. After U2 is authenticated by the key server, it must be sent the MTK which requires establishing the MUK. The current MUK and MTK cannot be used because this would allow U2 to decrypt previously transmitted location information that it may have received and stored.

In this case, distributing the MTK to U3 and U4 is the same as the case in which U2 left the group. For U1 and U2, a new K_{11}, $K_{11}*$, must be established. During the registration process U2 establishes K_{112}. Thus, $K_{11}*$ may be multicast to U1 and U2 protected individually by K_{111}

and K_{112}, or distributed point-to-point. Once $K_{11}*$ is in place, the new MTK, K_g*, may be multicast to U1 and U2 encrypted with $K_{11}*$.

There are several overhead components of concern when considering a re-keying protocol.

To quantify this overhead we define the following variables:

N : number of users in the group
h: height of the tree (h = log N)
K: encrypted key size
E: cost of encryption operation
D: cost of decryption operation
C_m*: overhead in terms of bytes for method m*
r_c*: ratio of users of class c in terms of percentage*
P: overhead for delivering a re-keying packet
c_{max}*: the maximum location classes or sub-groups*

Table 2 summarizes the messaging, storage and computation overhead for the basic LKH scheme.

Table 2. Summary of LKH overhead

Message overhead:

Join Event		Leave Event
Multicast (MTK distribution)	Unicast (MUK distribution)	*(2h-1)K*
(2h-1)K	*(h+1)K*	

Storage overhead:

Key Server	Member
(2N-1) K	*(h+1) K*

Computation Overhead:

Join Event		Leave Event	
KDC	Member	KDC	Member
3hE	*(h+1)D*	*2hE*	*hD*

The overhead for re-keying in this scheme is given by:

$$C_{LKH}(N) = 2K \log N + P \quad \textbf{(1)}$$

5.3. MIKEY-LKH applied to hierarchical coding

In this subsection we discuss the efficiency of re-keying for the different methods of distributing the hierarchically coded location information.

5.3.1. Coding by information class

In this scheme, the key server needs to maintain separate groups for each class of location information. A user must subscribe to multiple groups to receive more than one granularity of location information. When a user either leaves or joins a group, a new MTK must be distributed to all users that subscribe to the same classes of location information.

The cost of re-keying may be expressed as:

$$C_{class}(N) = \sum_{i=1}^{c_{\max}} r_i \left(2K\left(\sum_{j=1}^{i} \log\left(\left(\sum_{m=j}^{c_{\max}} r_m\right) \cdot N\right)\right) + iP\right) \quad \textbf{(2)}$$

In this scheme, each user of class i must store $\sum_{j=1}^{i}(h_j + 1)$ keys for each level of location information it wishes to receive, where h_j is the height of the key tree of group level j.

5.3.2. Coding by group

In this method, each group re-keys independently of all other groups. Therefore, if a user in group i leaves or joins, only those users in group i must be re-keyed. The re-keying procedures follow the basic re-keying protocol as for the basic LKH scheme.

The cost of re-keying may be expressed as:

$$C_{group}(N) = \sum_{i=1}^{c_{\max}} r_i (2K \log(r_i \cdot N) + P) \quad \textbf{(3)}$$

In this scheme, each user must store $1+h_i$ keys, where h_i is the height of the key tree of group i.

5.3.3. Nested hierarchy coding

For nested hierarchical coding a modified LKH can be used. The basic assumption for LKH is that the user knows the keys from the leaf (itself) up to the root, and the root key is the group key. In our definition, this is still true, but the modified LKH tree is not a balanced tree.

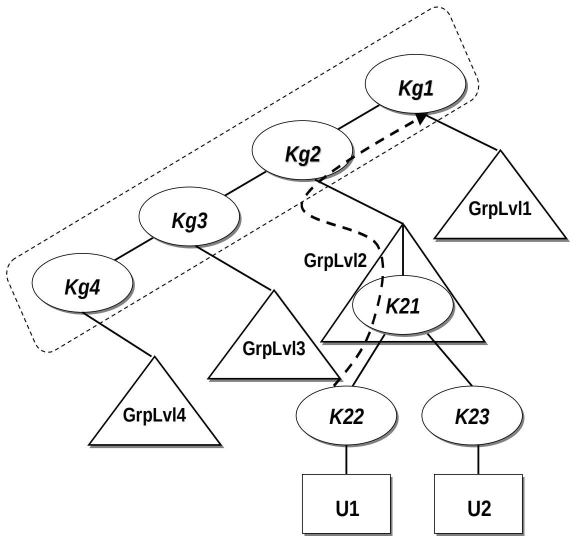

Figure 3. Key tree for hierarchical coding

The unbalanced tree structure is shown in Figure 3. The height of the tree is decided by the biggest group and is in general *O(logN)*.

When a user joins a group, the server adds the user to the appropriate group. The key server will change the keys on the path between the root and the new user. A similar re-keying takes place when a user leaves the tree.

Note that when a user with a low priority re-keys, only one group key, $K_{g1,}$ need to be changed. When a highest class user, class *c*, re-keys, all the group keys must be changed. This will be efficient if there are many more low priority users than high priority users.

In general, the re-keying overhead for the nested method is:

$$C_{nested}(N) = \sum_{i=1}^{c_{\max}} r_i (2K(\log(r_i \cdot N) + i) + P) \qquad \textbf{(4)}$$

In this scheme, a user in class *c* must store $c+h_c$ keys. This includes one for each class of location information it will access (*c*), plus the height of the tree of its group, h_c.

5.3.4. Flat encryption

To make the protocol more general the flat encryption scheme does not rely on information hierarchy and assumes a flat relationship among sub-groups. Each sub-group is still a balanced tree and the key server maintains the information subscription mapping among sub-groups.

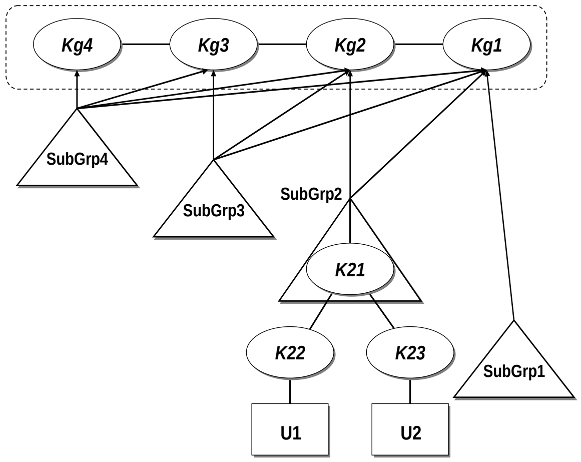

Figure 4. Key tree for flat coding

Figure 4 shows an example of the key tree with flat coding. Sub-group *c* can view the most precise location information.

When a user joins a group, the key server decides which sub-group to put the logical node for the user. The re-keying message will be handled by all the relative sub-groups. For example, if a user joins sub-group *c*, all the sub-groups keys need be changed.

The re-keying overhead for this scheme is:

$$C_{flat}(N) = \sum_{i=1}^{c_{\max}} r_i (K(2\log(r_i \cdot N) + \sum_{j=1}^{i} (c_{\max} - j + 1)) + P) \qquad \textbf{(5)}$$

In this scheme, each user must store $(c+h_c+1)$ keys where *c* is the class of the user, and h_c is the height of the *c*-class tree.

6. Performance evaluation

In the following subsections we evaluate the performance of each keying method. In these evaluations we assume the size of an encrypted key, K = 20 bytes, and that the protocol overhead, P=50 bytes.

6.1. Dynamic groups in basic LKH

Figure 5 shows the re-keying overhead in terms of Bytes/hour of the basic LKH scheme for two scenarios. In the first, the join/leave rate, λ = 100/hour. In the second, λ = 20/hour. These scenarios represent a system that is very dynamic (i.e., experiences a high join/leave rate) and one that is more stable (i.e., experiences a low join/leave rate).

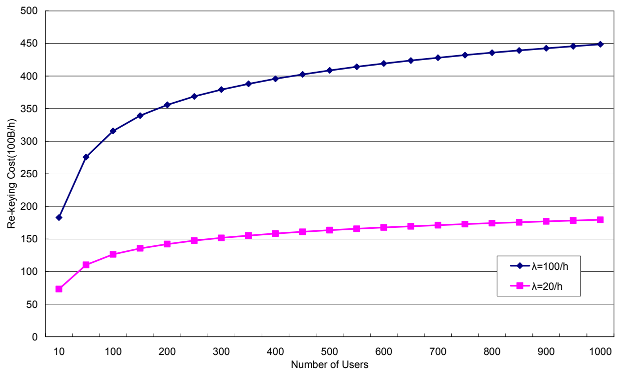

Figure 5. Basic LKH re-keying overhead

As expected, the growth rate of the overhead is below linear with respect to the increase in number of users.

6.2. Dynamic groups with hierarchical coding

Based on the four hierarchical coding schemes, we can deduct the average height of the tree with the different location preciseness levels. For this evaluation we assume that there a four classes of location information, and hence four classes of user. We assume the ratio among the different 4 sub-groups is $r_1{:}r_2{:}r_3{:}r_4$. We set λ, the join/leave rate to be 100/hour. The overall overhead for re-keying is then simply $\lambda C_{method}(N)$,where C_{method} is the byte overhead of re-keying for each method as defined in equations 2-5.

In Figure 6 we show the overhead for $r_1:r_2:r_3:r_4=30:10:8:5$. This represents a system in which many users have access to low granularity location information (e.g., country or state), and few have access to high precision location information.

From the figure, we can see that the three schemes (group, nested, and flat) that accommodate hierarchical coding of location information all have better performance than the direct application of LKH (coding by class) in terms of re-keying cost. The most efficient scheme in this respect is coding by group, i.e., each user group receives all data under a single key. This is intuitively most efficient as it requires re-keying only for users directly affected by the group membership change. With this ratio nested coding is more efficient than flat coding.

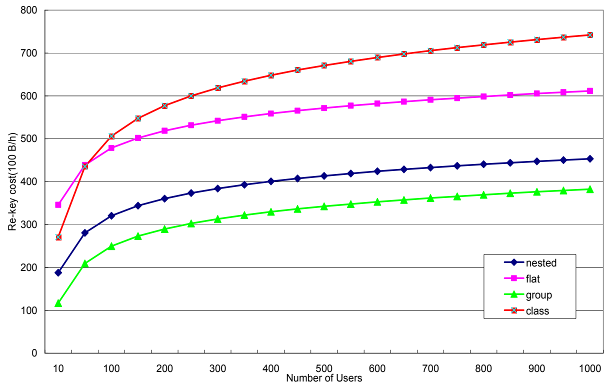

Figure 6. Hierarchical LKH re-keying overhead ($r_1:r_2:r_3:r_4$=30:10:8:5)

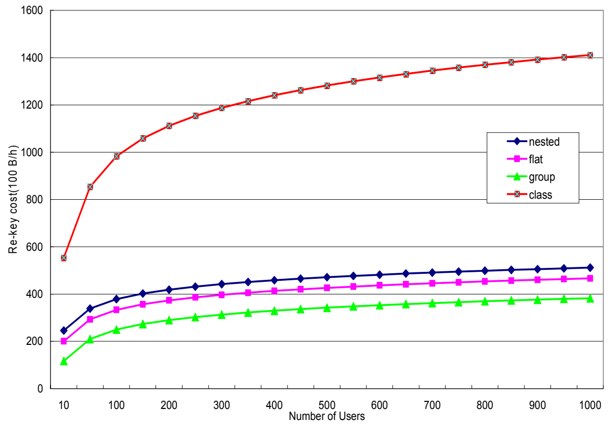

Figure 7. Hierarchical LKH re-keying overhead ($r_1:r_2:r_3:r_4$ = 5:8:10:30)

If we reverse the ratio $r_1:r_2:r_3:r_4 = 5:8:10:30$, as shown in Figure 7, we see higher increasing overhead for the basic LKH scheme because more users subscribe to multiple groups. This represents a system in which many users have access to high precision location information. In this case the three schemes that accommodate hierarchical encoding perform similarly.

7. Discussion

In this section we discuss the trade-offs of the various methods described above. This discussion is summarized in Table 3. In this table, + corresponds to an attractive characteristic, - corresponds to a drawback, and 0 is neutral.

The coding by class scheme is very effective if a single group is exists. However, it is not effective if multiple classes exist for four reasons. First, more multicast messages are required to deliver location information because users requiring more than one class of information are required to receive multicast messages on more than one group. Second, multiple groups of users must be managed from both a re-keying perspective and a data dissemination perspective. Multiple key trees are required because an independent tree is maintained for each class of data. Multiple groups from a data dissemination perspective are required because multiple groups exist, each receiving a different set of data. Third, the re-keying overhead is high as shown in Figures 6 and 7. Finally, user devices must store keys for each class of data they require.

Table 3. Summary of evaluation

Method	User Storage	Multi-cast groups	Re-key	Message delivery	Com-puting at end device	Flexi-bility
Class	-	-	-	-	-	+
Group	+	-	+	+	+	0
Nested	-	+	0	+	-	0
Flat	-	-	0	-	-	+

The coding by group solution is attractive for four reasons. First, it results in low re-keying overhead as shown in Figures 6 and 7. Second, the delivery of messages is efficient because users only receive the data they require. Third, each user receives the data at *and below* their maximum class level. This potentially simplifies service logic because the recipients of the high granularity data do not have to process it to derive the lower granularity information. For example, when a high priority user receives the *(x,y)* coordinates of a peer, it will also receive the state location of the peer and not have to derive it itself. Finally, the key storage requirements on each user device are low.

The main drawback of the coding by group solution is that there are *c* groups to be managed from both a re-keying and data dissemination perspective. From a re-keying perspective, a tree for each group is maintained. From a data dissemination perspective, each group is delivered a message containing only the data they request.

Like the coding by group solution, with the nested coding solution a user receives all the data at and below its maximum level. It has an additional benefit in that only a single multicast group must be maintained for data dissemination and re-keying. The nested solution has three main drawbacks. First, re-keying overhead may be high compared with the coding by group solution because in many cases groups must be re-keyed even if the change in membership occurred in a different group. Second, the processing overhead on high priority users is high because they must decrypt data *c* times to retrieve all class of data. Finally, key storage requirements in the user devices are high with this scheme compared with the coding by group solution.

The flat coding scheme is the most flexible from the user perspective. In this scheme a user may only join the groups they desire. Therefore, if a user requires location information at granularity 1, 2, and 4, but not 3, it does not have to join group 3. Likewise this method allows users more flexibility in deciding which data to disseminate. It provides the possibility that a user may receive all data at or below their maximum level if so desired. Its main drawbacks are the requirement to maintain multiple key trees and data dissemination groups, its re-keying overhead may be high, and the key storage requirements on the end systems may be high compared with the coding by group solution.

8. Conclusion

In this paper we presented an architecture to support flexible location based services. The architecture allows users to share their location information at different levels of granularity. The key idea behind the system is to hierarchically encrypt location information under different keys, and distribute the appropriate keys only to group members with permission to see the location information at that granularity.

We proposed extensions to the MIKEY key management protocol to enable efficient re-keying through the application of the LKH protocol. We tailored this protocol to four different hierarchical location information dissemination methods and analyzed their efficiency in several dimensions, including re-keying overhead, message delivery overhead, computation complexity, number of groups that must be managed, and flexibility from a services perspective.

In our future work we plan on evaluating implementation options for this architecture through experimentation.

References

[1] 3GPP; Functional Stage 2 description of Location Services (LCS): 3G TS 23.271: V6.8.0, (2004-2006).

[2] IETF GeoPriv working group, *http://www.ietf.org/html.charters/geopriv-charter.html*.

[3] Z. Lozinski, *Parlay/OSA - a new way to create wireless services*, White Paper of the Parlay Group, May 2003.

[4] M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," in *Proc. Ubicomp, LNCS 2498*, Springer-Verlag, 2002, pp. 237–245.

[5] G. Myles, A. Friday and N. Davies, "Preserving Privacy in Environments with Location-Based Applications", *IEEE Pervasive Computing*, 2003, 2(1), pp. 56-64.

[6] M. Gruteser and X. Liu, "Protecting Privacy in Continuous Location-Tracking Applications", *IEEE Security and Privacy*, Mar-Apr 2004, 2(2), pp. 28-34.

[7] M. Gruteser and D. Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *Proc. 1st Int'l Conf. Mobile Systems, Applications, and Services*, Usenix Press, 2003, pp. 31-42.

[8] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing", *Pervasive Computing, IEEE*, Jan-Mar 2003, 2(1), pp. 46-55.

[9] Q. He, D. Wu and P. Khosla, "Quest for Personal Control over Mobile Location Privacy", *IEEE Communications Magazine*, May 2004, 42(5), pp. 130-136.

[10] B. Schilit, J. Hong and M. Gruteser, "Wireless Location Privacy Protection", *IEEE Computer*, Dec. 2003, 36(12), pp. 135-137.

[11] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication", *ACM computing survey*, Sep.2003.

[12] D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architecture", *Internet Draft,* Internet Engineering Task Force, Sep 1998.

[13] C. Wong, M. Gouda and S. Lam, "Secure Group Communication Using Key Graphs," in *Proc. of SIGCOMM*, Vancouver, British Columbia, 1998, pp. 68-79.

[14] D. Balenson, D. McGrew and A. Sherman, "Key management for large dynamic groups: One-way function trees and amortized initialization", *Internet Draft*, Internet Engineering Task Force (Work in progress), August 2000.

[15] A. Perrig, D. Song and J. D. Tygar, "ELK, a new protocol for efficient large-group key distribution," in *Proc. of the IEEE Symposium on Security and Privacy*, May 2001, pp. 247-262.

[16] M. Baugher and B. Weis, etc., "The Group Domain of Interpretation", *rfc3547*, IETF Standard, July 2003.

[17] H. Harney and U. Meth, etc., "GSAKMP", *draft-ietf-msec-gsakmp-sec-06.txt*, IETF draft, June 2004.

[18] J. Arkko and E. Carrara, etc., "MIKEY: Multimedia Internet KEYing", *rfc3830*, IETF standard, August 2004.

[19] 3GPP; Multimedia Broadcast/Multicast Service (MBMS); Architecture and Functional Description: 3G TS 23.246 V6.7.0 (2005-2006).

[20] 3GPP; 3G Security; Security of Multimedia Broadcast/Multicast Service: 3G TS 33.246 V6.3.0, (2005-2006).